Behavioral cloning

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road

Files included

a) model.py : contains the generator, train , test splitting and calls the model
b) network.py : contains the actual model
c) model.h5 is the trained model file
d) drive.py : is the python file which connects to the simulator and communicates the steering angle to it as predicted by the model

1. Simulator data
   I did not generate data on my own. I found that my driving was kind of erratic and Driving from side to side which I thought would result in a bad model. I used the model shared by Udacity and augmented it. I used the left and right images and added/subtracted a correction factor which would make the car learn to keep to the center. I also flipped the images so that it would not have a left turn bias. I took care to reverse the sign of the steering measurement when the image Is flipped.

2. Network
   I started off with Lenet. Which didn't do much except get the car moving, it was not Able to get past the first curve even. I later switched to the NVIDIA's model for self driving car.

   The lamda layer does a normalization

   A summary of the model is as shown below

   Convolution layer

```
Layer (type)                Output Shape        Param #    Connected to
====================================================================
lambda_1 (Lambda)             (None, 160, 320, 3)   0        lambda_input_1[0][0]
____________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 78, 158, 24)   1824      lambda_1[0][0]
____________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 37, 77, 36)    21636
convolution2d_1[0][0]
____________________________________________________________________
```

```
convolution2d_3 (Convolution2D)  (None, 17, 37, 48)   43248
convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 15, 35, 64)   27712
convolution2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 13, 33, 64)   36928
convolution2d_4[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 27456)        0           convolution2d_5[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 100)          2745700     flatten_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 50)           5050        dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 10)           510         dense_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 1)            11          dense_3[0][0]
====================================================================================================
```

The final output predicts the steering angle

The model was converging pretty well in the first 3 epochs. Any more epochs and the validation loss would increase. Hence the number of epochs were kept at 3

```
Epoch 1/3
38592/38568 [==============================] - 158s - loss: 0.0211 - val_loss:
0.0162
Epoch 2/3
38592/38568 [==============================] - 156s - loss: 0.0134 - val_loss:
0.0150
Epoch 3/3
38592/38568 [==============================] - 156s - loss: 0.0114 - val_loss:
0.0130
```

There was no attempt at introducing any regularization by introducing dropouts as the model seemed to behave well enough as it is. The training / testing split that was done initially is the only thing done to prevent over fitting.

3. Training and validation
   I used a 80:20 split between training data and validation. I used a generator for training data and validation data. The generator greatly reduced the RAM requirements of the system as a whole.

4. Used the generated model to drive around the track
   I had used a steering correction of 0.3 for the non center images, with this, the car was driving in a zig zag manner. When I left the simulator running for multiple laps,

I found that the zig zag would start and would increase in magnitude till the car would eventually go off course.

I hypothesized that this was because of a) Large correction factor or b) imbalance in the number of left/ right /straight steering data

Changing the correction factor to 0.25 reduced the zig zag manner. I did not spend time rebalancing the number of left/right /straight steering data

5. Further testing
   If I run the model on the same track but in reverse (clock wise), the car sometimes goes off the track. Still trying to figure out why this is.
   The model performs horribly on track 2. This at least is understandable as the track 2 has different kinds of lane lines, shadows, much curvier roads

6. Further improvements
   I plan to refine the model by including some of the track 2 images.
   Figure out why the clockwise lap of the track 1 fails
   Introduce cropping and scaling to make do with lesser and necessary data
   Introduce drop outs to decrease over fitting